VI & VIC

Nr. 2 - 2. årg. - Juni-juli 1983 Pris 17,95

Garabage collection er en tidrøver, der kan sættes ud af spillet med to subrutiner

Det er ikke al den hukommelse, som ikke bruges i VIC, der er tom. Faktisk slettes indholdet af streng-området ikke før, der virkelig er brug for pladsen. Så længe der er tom plads andre steder, bruges den først.
I praksis foregår det på den måde, at variabeltabellen indeholder oplysninger om, hvor strengene befinder sig (hvor de starter), og hvor lange de er. Hvis man f.eks. har en streng-variabel, A$, der hedder ''VI & VIC'', gemmes der i variabeltabellen oplysninger om, hvor i hukommelsen, variablens indhold er gemt, og der er en oplysning om, at den er 8 karakterer lang. Maksimum er 256 karakterer.
Hvis man herefter ændrer A$, så den får indholdet ''BRUGERBLAD'', vil det af tabellen fremgå, at A$ er 10 karakterer lang, men den ligger ikke det samme sted, som den oprindelige A$ - den ligger i et nyt område af RAM-hukommelsen. Det er samtidig registreret, at den del af hukommelsen, som var brugt til ''VI & VIC'' nu gerne må bruges til noget nyt.



FORTH ord har både hoved og krop

Læs side 6

Sådan lægges VIC-programmer ind i C 64

Læs side 8

Monitor-program til Commodore 64

Læs side 9

Fortsat fra forsiden

Det sker imidlertid ikke, før computeren virkelig har brug for det - og det vil sige, når hele hukommelsen er ved at være brugt op. Så går computeren på GARBAGE COLLECTION - "spildsamling" - og frigør på den måde den ledige plads.

fremmest ved deling af de strenge, der indeholder oplysninger om de enkelte poster, og det sker, når computeren skal danne den linie, som skal skrives.
Her laver man nemlig først én streng, som indeholder datoen - derefter ændres strengen, så den også indeholder teksten til posten, og den ændres, så den får fra 1 til 3 talkolonner rigtigt på plads - og hver gang linien ændres, kopieres der nye strenge ned i streng-området.

```
20 POKE251,PEEK(55):POKE252,PEEK(56):POKE55,
   PEEK(51):POKE56,PEEK(52):X=FRE(0):RE
TURN
21 POKE55,PEEK(251):POKE56,PEEK(252):RETURN
```

Tager lang tid

Problemet med garbage collection er tiden. Selv om en computer arbejder meget hurtigt, kan en sådan frigørelse af ledig RAM-hukommelse tage ganske lang tid - i ekstreme tilfælde så lang tid, at man får det indtryk, at programmet er gået i stå, eller computeren simpelthen er faldet i søvn.
Omfanget af garbage collection-problemet afhænger dels af, hvor megen hukommelse der skal gennemsøges for at frigøre ledig RAM, dels af, hvor mange streng-operationer, man arbejder med.
Et eksempel er givet i det regnskabsprogram, som vi præsenterer i dette nummer af VI & VIC. Her foregår ganske mange streng-operationer, når en kassejournal eller et momsregnskab skal skrives ud. Det sker først og

Har man ikke mange variabler i hukommelsen, betyder det, at man kun har et lille område at gennemsøge for at finde ledig RAM, men har man mange, kan sagen blive alvorlig.
I dette konkrete tilfælde betød det, at udskrivningen af en enkelt side kan tage op mod en halv time - hvor nogle få minutter ville være mere rimeligt.

Løsningen

Men heldigvis kan mange problemer løses - og også dette. Trick'et består i, at man bilder computeren ind, at streng-området er meget mindre end det i virkeligheden er - og derfor bliver der et mindre område at gennemføre garabage-collection i. Helt konkret sætter man i subrutinen i linie 20 pointerne, der viser, hvor hukommelsen ender (top of memory) til det sted, hvor strengene ender (bottom of strings). Strengarealet ligger nemlig i toppen af hukommelsen, og med denne manøvre undgår man garbage collection i hele det område, hvor de oprindelige strenge befinder sig. Resultatet: Lynhurtig udskrivning.
Men man skal huske at sætte pointerne tilbage igen, når man er færdig med udskrivningen - og det sker i subrutinen i linie 21. De oprindelige værdier er gemt i hukommelsesplads 251 og 252 - pladser, som er ledige i både Commodore 64 og VIC-20.

VIC

VI & VIC udgives af commodore data a/s,
Bjerrevej 67, 8700 Horsens.
Redaktion og tilrettelæggelse:
Baghuset Grafik/Design, Vejle.
Ansvarshavende redaktør: Steen Venbjerg.
Annoncer: Lise Tylvad,
Æblehaven 7, 7100 Vejle.
Tryk: Baghuset Grafik/Design, Vejle.
Redaktion og ekspedition:
Kornager 29, 7100 Vejle.
Giro nr. 6 17 39 93

ISSN 0108-2213

Brug ikke REPLACE på disketterne

Det er en snild funktion på VIC diskettestationen, at man ved hjælp af det lille a, omkranset af en ring kan gemme programmer og datafiler under et allerede eksisterende navn - hvor det gamle indhold så erstattes af det nye.
Men vi må advare mod at bruge denne mulighed, for der er set eksempler på, at der gik kludder i de pointere, der fortæller, hvor de forskellige ting er gemt, og ét mistet program er tilstrækkelig grund til helt at undgå REPLACE-funktionen. Og den kan undgås meget enkelt.
Problemer opstår, fordi der i virkeligheden ikke sker det, man umiddelbart kunne tro, når programmøren har revideret programmet og ønsker at gemme den nye version og slette den gamle. Så kan han skrive SAVE " @ PROGRAM",8 - hvilket vil betyde, at det nye program er gemt under samme navn, som det gamle - i hvert fald indtil en vis grænse. Det gamle program slettes nemlig ikke. Det nye program bliver gemt i et frit område på disketten, og det registreres, at det gamle program ikke gælder mere. Gentages proceduren mange gange - og det kan der let være grund til at gøre under udviklingen af et nyt program, når man til sidst frem til en situation, hvor der simpelthen ikke er mere plads på disketten.



I den situation begynder diskettestationen at udføre den såkaldte ''Gabage collection'', hvilket vil sige, at der skal ryddes op i de kasserede programmer. Og det er her, det kan gå galt. Det er en svaghed i systemet, som den enkelte bruger ikke kan afhjælpe - men man kan som nævnt helt undgå REPLACE-funktionen, og her er vort bud på, hvordan det gøres.

De tre programlinier, vi gengiver her ved artiklen kan stå som de sidste linier i det program, man arbejder med -og hver gang, man ønsker at gemme sit program og slette den gamle version, skriver man blot RUN 63998.
Denne instruktion vil gemme programmet under navnet ''DUMMY'', hvorefter den sletter det oprindelige program. Men da man jo ønskede at køre videre med det samme navn, døbes det nye program om, så det nu får det gamle navn igen. Voila. I linie 63999 kontrolleres det, om programmet er kommet rigtigt ned på disketten.
Læg mærke til, at linie 63997 indeholder END. Det er gjort for at sikre, at man ikke ved kørsel af programmet uforvarende kører ud i denne lille REPLACE-funktion. Er man sparsommelig med pladsen, kan END skrives

```
63997 END
63998 OPEN1,8,15:SAVE"DUMMY",8:PRINT#1,"S0:PROGRAM":PRINT#1,"R0:PROGRAM=DUMMY":C
LOSE1
63999 VERIFY"PROGRAM",8
```

Programmører må lægge vægt på at gøre deres programmer overskuelige

Fra Rasmus og Sven Bertelsen i Holte har vi modtaget et hjertesuk, der drejer sig om brugernes mulighed for at forstå indholdet af et program, som man ikke selv har skrevet - men som man gerne vil sætte sig ind i.
Blandt andet efterlyses i mange programmer REM-linier og en mere konsekvent brug af SUB-rutiner, og selv om det fylder lidt mere, er det også et ønske at få flere mellemrum mellem kommandoerne i den enkelte programlinie.

Der er ingen tvivl om, at denne kritik er helt berettiget, og her på redaktionen vil vi gerne slå et slag for anbringelsen af instruktive REM-sætninger, der kan gøre forståelsen af programmerne lettere.
REM-sætninger kan anbringes hvor som helst i et program, hvor de har den funktion, at de kan læses, når programmet listes - men de påvirker ikke programafviklingen. Når programmet når en REM-instruktion, springes resten af programlinien ganske enkelt over.
Efter REM kan man skrive, hvad som helst, men man må holde sig fra de grafiske tegn, fordi de ved en listning vil blive omskrevet til diverse BASIC-instruktioner. Prøv f.eks. at skrive en programlinie:

10 REM TEST
- men hold SHIFT-tasten nede, mens ordet TEST skrives, så der kommer grafiske tegn på skærmen. Når linien LISTes, står der 10 REM LETVALREADLET.

Logisk plan

Der er næppe tvivl om, at den første forudsætning for, at et program kan få en klar og logisk opbygning er, at programmøren som udgangspunkt for sit arbejde har haft en klar og logisk plan med sit arbejde.
Selv om enhver VIC-bruger naturligvis helst går direkte til tastaturet, når et program skal skabes, kan det alligevel ofte betale sig at gå omvejen over et stykke papir, hvor man laver et rutediagram, som programmet skal følge. Det behøver slet ikke være noget fornemt dokument, hvis det kun skal bruges af en selv - bare nogle firkantede kasser, hvor man har skrevet op, hvilke funktioner, forskellige dele af programmet skal være i stand til at udføre. Alene det at udarbejde et sådant diagram kan ofte være med til at give en større forståelse for, hvilken struktur, programmet skal have. Man vil f.eks. blive klar over, om de forskellige blokke af programmet kommer til at fungere som subrutiner, der kal-

Fortsættes næste side

des af de andre dele af programmet.

Subrutine

Når man går i gang med at beskrive indholdet af de enkelte programdele, vil man normalt blive klar over, at mange af operationerne gentages forskellige steder - og her er det helt oplagt at benytte subrutiner.
Skal man f.eks. have én lyd, der kalder på brugerens opmærksomhed i forbindelse med input og én lyd, der gør opmærksom på fejl, vil det være naturligt at programmere disse én gang for alle i et par subrutiner, der kan kaldes fra en hvilken som helst del af programmet.
Lav fra starten en liste over, hvilke subrutiner, der findes og lav hellere en subrutine for meget end én for lidt.
Det er en stor fordel at bruge subrutiner, fordi det gør arbejdet med programmering lettere. Når først subrutinen er skrevet ind, skal man ikke gentage det samme arbejde igen og igen. Subrutinerne giver desuden større klarhed i programmet - ikke mindst, hvis man forsyner hver subrutine med en REM-sætning i starten, hvor rutinens funktion er beskrevet, eller blot har fået et navn, man kan genkende.

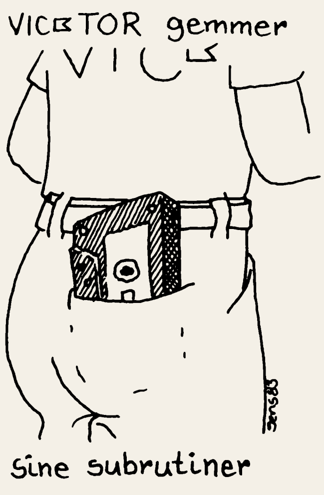
Sparer plads

Endelig vil der normalt være plads at spare ved at bruge subrutiner. Ved meget korte subrutiner er der måske ikke noget sparet før tredie gang, der er brug for den samme rutine i programmet - for adresseringen tager plads - og der skal også være plads til en RETURN til afslutning af rutinen.
Har man dette forhold i tankerne, kan der spares lidt ekstra plads ved at placere subrutinerne i programmets lave linienumre, så der kun skal adresseres til 1- eller 2-cifrede linienumre. Det er småting, men en byte sparet kan i en snæver vending blive afgørende for, om programmet kommer til at køre.
En idé til efterfølgelse kan være at lade ethvert program starte med at definere de konstanter, der er brug for i programmet - og derefter hoppe over afsnittet med subrutiner for at tage fat på det egentlige program fra linie 1000 - eller hvad man måtte finde passende.

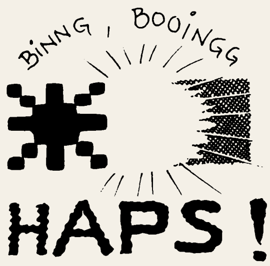
Sådan kan du gøre livet lettere for dig selv: Lav subrutine-bibliotek

Subrutiner er en af vejene frem mod en lettere og mere gennemskuelig programmering, men det kan være irriterende at skulle starte ethvert programmeringsarbejde med at taste de gammelkendte subrutiner ind. Så hvorfor ikke gøre det én gang for alle: Lav et subrutinebibliotek, hvor du kan hente de ønskede funktioner.

Er man den lykkelige ejer af en Programmers Aid, er det ingen sag at flytte rutiner fra ét program over i et andet. Ved hjælp af RENUMBER og MERGE funktioner kan eksisterende programmer kombineres med hinanden, men selv om man »kun« har en kassettebåndoptager til sin rådighed, er man ikke helt fortabt i denne sammenhæng.
Måden at gøre det på, kan være den enkle, at man i ét program gemmer de subrutiner, som man tror, man kan få brug for i nye sammenhænge. Lad et sådant program være starten på ethvert andet program, man vi lave I stedet for at starte programmeringen fra bar bund, starter man med at indlæse sit subrutine-program - hvorefter man bygger sit nye program ovenpå.
Hvis det efter endt arbejde viser sig, at der er nogle af subrutinerne, som man slet ikke fik brug for, er det jo ingen sag at slette dem fra det nye program - og det er nu én gang lettere at slette et antal programliinier end at skulle sidde og taste dem ind igen.
For at få det fulde udbytte af et sådant subrutine-bibliotek, er det naturligvis nødvendigt hele tiden at føre sit biblioteksprogram a jour. Så hver ny subrutine skal altså skrives to gange - men så skulle det heller ikke være nødvendigt at skrive de samme linier nogen sinde mere.



Et spil til VIC, hvor det gælder om at hapse uden selv at blive det



Det spil, vi her gengiver programlistningen til, bærer navnet HAPS, og det er netop, hvad det drejer sig om. Ved spillets start skal man bestemme sig for, hvor mange deltagere, der skal være med til spillet - og hvor mange omgange, der skal spilles. der kan vælges mellem 5 forskellige sværhedsgrader.

Herefter fordeler der sig en blanding af forskellige tegn over skærmen, som man skal »hapse« med en markør, der kan styres i de fire retninger. Men kommer man til at møde en stjerne, bliver man selv hapset og spillet er ude. God fornøjelse.

```
10 PRINT"█"
20 DIMPL(6),R(5)
30 FORY=1TO5:FORX=1TO6:Z(X,Y)=0:NEXTX:NEXTY
40 C=30720:TB=0:TS=0
50 POKE36879,239
60 CP=0:GOTO810
70 PRINTTAB(4)"██████AF A.RASMUSSEN"
90 CP=1
100 FORT=1TO2000:NEXTT
110 PRINT"█"
120 PRINTTAB(5)"█INSTRUKTION"
130 PRINTTAB(5)"───────────"
140 PRINT"█DU HAR 2 MIN.":PRINT"TIL AT SCORE "
150 PRINT"██RETNING:":PRINT"█CRSR↑=VENSTRE
   ":PRINT"CRSR←=HØJRE":PRINT"F5=OP"
160 PRINT"F7=NED":PRINT"█████"
170 PRINTTAB(14)"████SCORING:"
180 PRINTTAB(14)"█O█=1"
190 PRINTTAB(14)"█*█=2"
200 PRINTTAB(14)"███=3"
210 PRINTTAB(14)"███=5"
220 PRINTTAB(14)"██=10"
230 PRINT"█DU ER: ██ █"
240 PRINT"█RAM IKKE EN █*█ SÅ":PRINT"BLIVER DU █HAPSET."
250 PRINTTAB(3)"██TRYK PÅ EN TAST"
260 GETA$:IFA$=""THEN260
270 PRINT"██HVOR MANGE RUNDER(1-5)"
280 INPUTRN:IFRN<1ORRN>5THENPRINT"UH?":GOTO270
290 PRINT"█HVOR MANGE SPILLER(1-6)"
300 INPUTPN:IFPN<1ORPN>6THENPRINT"UH?":GOTO290
310 FORR=1TORN
320 FORP=1TOPN:PRINT"██SPILLER #";P
330 PRINT"█SVÆRHEDSGRAD?"
340 PRINT"TRYK 0,1,2,3 OG 4";
350 INPUTS
360 IFS>4ORS<0THENPRINT"UH?":GOTO340
370 PRINT"███HØJEST SCORE :";TB:PRINT"█SVÆRHEDSGRAD:";SL
380 PRINT"█SPILLER #";PB
390 FORT=1TO3000:NEXTT:PRINT"█"
400 DEF FN A(L)=INT(RND(1)*L)+7702
410 FORF=1TO40-2*S:D=FNA(483)
420 POKED,87:POKED+C,0:NEXTF
430 FORF=1TO25:D=FNA(483)
440 POKED,81:POKED+C,3:NEXTF
450 FORF=1TO10+4*S:D=FNA(505)
460 POKED,42:POKED+C,4:NEXTF
470 FORF=1TO19:D=FNA(483)
480 POKED,90:POKED+C,7:NEXTF
490 FORF=1TO14:D=FNA(483)
500 POKED,83:POKED+C,2:NEXTF
510 FORF=1TO9+S:D=FNA(505)
520 POKED,65:POKED+C,1:NEXTF
530 B=7932
540 TI$="000000"
550 PRINT"█TID:";120-INT(TI/60);"█ "
560 IFTI/60>=120THENGOTO930
570 POKEB,160:POKEB+C,6
580 IFB<7636THENB=8229+B-7635
585 IFB>8229THENB=7636+B-8230
590 IFPEEK(197)=31THENH=190:DR=-1:GOTO630
600 IFPEEK(197)=23THENH=200:DR=1:GOTO630
610 IFPEEK(197)=55THENH=210:DR=-22:GOTO630
620 IFPEEK(197)=63THENH=220:DR=22
630 POKE36878,15:POKE36876,H
640 FORT=1TO30-5*S:NEXTT
650 POKEB,32:B=B+DR
660 SC=0
670 IFPEEK(B)=42THENGOTO790
680 IFPEEK(B)=87THENSC=1:GOTO740
690 IFPEEK(B)=81THENSC=2:GOTO740
700 IFPEEK(B)=90THENSC=3:GOTO740
710 IFPEEK(B)=83THENSC=5:GOTO740
720 IFPEEK(B)=65THENSC=10:GOTO740
730 GOTO760
740 TS=TS+SC
750 POKE36878,15:POKE36876,160+PEEK(B)
760 FORT=1TO30-5*S:NEXTT
770 PRINT"██SCORE=";TS
780 GOTO550
790 POKE36878,15
800 FORPI=1TO40:POKE36876,180-PI:NEXTPI
810 PRINT"██████████ █  █ █ █    █ █    █ █    █"
820 PRINT"██ █  █ █ █ █  █ █ █ █  █ █ █ █"
830 PRINT"██ █  █ █ █ █  █ █ █ █  █ █ █ █"
840 PRINT"██    █ █    █ █    █ █    █"
850 PRINT"██ █  █ █ █ █  █ █ █ █       █ █"
860 PRINT"██ █  █ █ █ █  █ █ █ █       █ █"
870 PRINT"██ █  █ █ █ █  █ █ █ █    █    ██"
880 IFCP=0THEN70
890 POKE36878,0:POKE36876,0
900 FORT=1TO2000:NEXTT:PRINT"█"
910 PRINT"███DU KLAREDE";INT(TI/60)-3;"█ SEKUNDER"
920 GOTO970
930 POKE36878,15:FORAC=1TO80:POKE36876,210-AC:NEXTAC
940 POKE36876,0:POKE36878,0
950 PRINT"█"
960 PRINT"██████ ██..TIDEN ER UDLØBET.."
970 PRINTTAB(6)"███SCORE=";TS
980 Z(P,R)=Z(P,R-1)+TS
990 IFTS>TBTHENTB=TS:SL=S:PB=P
1000 PRINT"██RUNDE #:";R;"█"
1010 FORX=1TOPN
1020 PRINT"█SPILLER #";X;":";Z(X,R)
1030 NEXTX
1040 TS=0:DR=0:H=0:PRINT"█"
1050 NEXTP:NEXTR
1060 END
```

FORTH-ordene har både hoved og krop

FORTH sproget består af fem nøgleelementer: Ordlisten, interpreteren, stackene, assembleren og extern hukommelse.
Lad os prøve at se nærmere på ordlistens opbygning. Sidst definerede vi ordet READ.

: READ PADDLE C@;
FORTH gemte dette ord i ordlisten, så når vi for fremtiden skriver READ, vil vi få status af vores joystick eller paddleport anbragt øverst på stacken, men lad os se, hvordan ordet read egentlig ser ud i vores ordliste. Lad os bruge assembler format for overskuelighedens skyld.

```
.BYTE $84,'rea',$C4 -- NFA
.WORD BALL       <-- LFA
.WORD DOCOL      <-- CFA
.WORD PADDLE     <-- PFA
.WORD C-FETCH
.WORD SEMIS
```

Hoved og krop

Et FORTH ord består af et hoved og en krop. Hovedet indeholder navnet, et linkfelt som FORTH bruger, når ordlisten afsøges, samt et felt, der fortæller hvilken type ord det er, højniveau eller assembler. Kroppen er den egentlige definition og i et højniveau ord indeholder kroppen en række to-byte adresser på de definerede ord. I hovedet finder man først navnefeltet, i FORTH kalder man den storrage adresse, hvor navnefeltet starter for navne-felt-adressen NFA, disse forkortelser er standard i FORTH, så lad os bruge dem.

De enkelte bytes

Den første byte i navnefeltet for READ var $84, lad os se, hvad de enkelte bits bliver brugt til:

```
7 6 5 4 3 2 1 0
      I I I I I  Længdetæller = antal bogstaver i navnet.
    I            SMUDGE bit.
  I              IMMEDIATE bit
I                Altid »1«
```

Der er fem bits til at angive længden på navnet, et FORTH navn kan altså være fra 1 til 31 karakterer i længden.
SMUDGE betyder snavs, og FORTH bruger denne bit til at »tilsmudse« et navn, så interpreteren ikke kan finde det i ordlisten. Denne bit vil være »1«, medens et ord bliver defineret for at undgå, at et ord fejlagtigt refererer til sig selv.
IMMEDIATE bitten fortæller interpreteren, at et ord skal udføres som en kommando med det samme, selv om det under kompileringen findes i definitionen af et andet ord.

Længdetæller

Nu er 31 karakterer jo sjældent nødvendige for entydigt at definere et navn, det ville bruge alt for meget af vores dyrebare storrage. Man kan ved hjælp af ordet WITH fortælle FORTH, hvor mange karakterer af et navn, man ønsker at gemme i ordlisten. Hvis vi definerer følgende ord: DETTE-ER-ET-MEGET-LANGT-ORD og har sat WITH til 10, vil længdetælleren angive 27 karakterer, men kun de 10 første karakterer af navnet vil blive gemt. Derfor sætter vi bit 7 af den sidste karakter til »1«, således at vi kan finde enden på navnet.

Efter navnet kommer link-felt-adressen LFA, i eksemplet adressen på ordet BALL. LFA er adressen på det foregående ord i ordlisten, og vi forudsætter her, at vi umiddelbart før READ har defineret ordet BALL. LFA bruges til at søge gennem ordlisten fra det ene navn til det næste. Ordlisten bliver således afsøgt bagfra startende med det sidste definerede ord, når man når en LFA med adressen $0000, har man nået det første ord i listen, og man fandt altså ikke det ord, man søgte efter.

Højniveau

Det næste, vi møder, er code-field-adressen CFA. CFA er adressen på den maskinsprogsrutine, man udfører, når man skal eksekvere et ord. Et ord er et højniveau ord, når CFA indeholder adressen på rutinen.
DOCOL rutinen vil tage indholdet af FORTH's Interpreter-Pointer IP og gemme det på returstacken. Derefter vil DOCOL tage den første parameter fra READ og anbringe den i IP.

Læs mere om FORTH

Gode artikler om FORTH:
DOCTOR DOBS JOURNAL Volume 3, issue 5 (may 78).
BYTE, August 80.

Gode bøger at starte med:
Starting FORTH by BRODIE
KITT PEAK PRIMER, BY STEVENS

Bøgerne og særtryk af artiklerne kan skaffes gennem FORTH Interest Group.

FORTH fylder mindre

Adressen på et ords parameterliste kaldes i FORTH Parameter-field-adressen PFA. Parameterlisten er en række to-byte felter, der indeholder CFA af de definerede ord, det første er CFA af PADDLE, det næste er CFA af C@ o.s.v. IP indeholder altså nu CFA af det første ord i definitionen af READ, nemlig PADDLE, og interpreteren vil nu begynde at udføre dette ord. Når FORTH ord eksekveres, kender FORTH altså CFA af de definerede ord, og kan derfor gå direkte til ordet uden at skulle lede ordlisten igennem først, for at finde navnet. Det sparer tid, for FORTH eksekverer da også mange gange hurtigere end Basic.
CFA fylder to bytes, medens et subrutine kald i assembler fylder tre bytes, det sammen med parameteroverførslen via den parameter stack gør, at store komplekse programmer fylder mindre i FORTH, end hvis de blev skrevet i assembler.

Lavtniveau

Lad os se, hvordan et lavtniveau ord skrevet i assembler vil se ud i ordlisten, vi kan tage ordet C@, som også blev brugt i definitionen af READ:

```
.BYTE $82,'C',$C0 <--NFA
.WORD TOGGLE      <--LFA
.WORD * + 2       <--CFA
LDA (0,X) Fetch byte addressed by top of stack
STA 0,X        Store on stack
STY 1,X
JMP NEXT
```

CFA peger til kode i selve ordet, som således indeholder den maskinsprogrutine, man udfører, når ordet eksekveres. En maskinsprogrutine afsluttes med JMP NEXT, et hop tilbage til interpreteren, som vil eksekvere det næste ord, her ordet SEMIS, maskinsprogrutinen for semicolon.

Man kan ikke fortælle alt om FORTH i et par små artikler, men har I fået blod på tanden, og ønsker I at vide mere om dette sprog, så følger der en litteraturliste efter denne artikel.

Tilbage er der så problemer med at skaffe sig et FORTH system. Til de fleste micro computere kan maskinleverandøren i dag skaffe et »nøglefærdigt« system, og hvis det passer til ens maskinconfiguration og pengepung, er det en nem start.

FORTH-forening

FORTH er PUBLIC DOMAIN, d.v.s. ingen kan tage patent på FORTH og alle oplysninger er tilgængelige. Så er man interesseret i at være med til at opbygge og udvide dette sprog, kan man melde sig ind i FORTH Interest Group, en seriøs forening dannet i 1978, med det formål at udvide kendskabet til FORTH programmering. Foreningen tæller i dag mere end 3500 medlemmer og udgiver bladet FORTH DIMENSIONS. Foreningen udgiver desuden på nonprofit basis forskellige publikationer, som er nødvendige for brugen af FORTH, f.eks. beskrivelse af standard ord, assembly liste af systemet m.m. Medlemskab koster i øjeblikket 27$ om året, det inkluderer bladet FORTH DIMENSIONS frit tilsendt med luftpost.

Adressen er:
FORTH Interest Group
P.O. Box 1105
San Carlos, CA 94070
U.S.A.

Sådan bærer man sig ad, hvis programmet selv skal ændre programmet

Erland Thölen og Henrik Laulund vil gerne have et program til at ændre eller slette linier »i sig selv« under kørslen. De har prøvet at lade programmet skrive:
100 DATA 3,3
på en helt blank skærm, hvorefter de har POKEt på følgende måde: 198,6 -631,19, 632,13, men programmet vil ikke godtage sætningen, medmindre programmet stoppes lige bagefter.

Og her er vi netop ved sagens kerne. Teknikken er nemlig god nok - men der mangler lige det sidste for at det kan fungere. Det er nemlig tastaturets bufferlager, der er benyttet - og den bliver ganske rigtigt først aftastet, når programmet standser (eller hvis man når en INPUT- eller GET-instruktion).

Derfor er løsningen ganske enkelt, at man lader programlinien ende med END, hvorefter programmet standser og den ønskede linie, der er printet på skærmen, lægges ind i programmet.

To problemer

Det giver imidlertid to nye problemer (som vi dog kan løse). For det første skal programmet jo gerne i gang igen - og for det andet vil man gerne bevare eventuelle variabler i programmet.

Det første problem er lettest at løse. Man lader blot programmet PRINTe den nye ønskede linie - og nedenunder skal programmet skrive "GOTO XXX", hvor XXX er det linienummer, hvor programmet skal fortsætte, efter at den nye linie er lagt ind i programmet.

Til de POKE-ordrer, der er nævnt ovenfor, skal derfor tilføjes endnu én, nemlig 633,13.

Den nærmere forklaring på denne teknik er, at man med 198,6 gør VIC opmærksom på, at der ligger et antal (6) karakterer i tastaturet og venter på at blive udført. Der er plads til ti karakterer i VIC-20 tastaturbufferen, og de har nummeret 631-640 - og de udføres i den rækkefølge. Værdien 19 står for "HOME", der bringer markøren op i venstre hjørne af skærmen, hvorefter værdien 13 udløses. Den står for RETURN og lægger altså den nye linie ind i programmet. Herefter skulle markøren gerne stå og blinke i linien "GOTO XXX", men den kommer ikke til at blinke længe, for den næste RETURN udløses omgående, hvorefter programmet fortsætter i linie XXX. Det er i øvrigt den samme teknik, der er anvendt i ét af de regneopgaveprogrammer, der var i sidste nummer af VI & VIC.

Variablerne

Det andet problem ved at ændre programmet, mens det kører, er, at samtlige variabler vil blive slettet. Det sker hver gang programmet ændres - og det sker også, når man skriver RUN. Det er svært at forestille sig et program helt uden variabler, så hvis man vil lave om på sit program, mens det kører og alligevel bevare variablerne, må man finde på én eller anden måde at gemme dem.

Man kan f.eks. gemme dem et sted, hvor de sikres imod at blive slettet, og det kunne eksempelvis være kassettebufferen. Her må alle variablerne lægges ned - og de må bagefter hentes frem igen. Det skal foregå med POKE og PEEK, så det vil jo ikke ligefrem peppe programafviklingen op. Men det kan lade sig gøre, og det er jo det vigtigste.

Det er ikke så ligetil, men VIC-programmer kan godt komme ind i 64'eren

Det første problem, nybagte Commodore 64-ejere beskæftiger sig med, er »hvordan får man sine VIC-programmer til at køre i 64'eren«?
Det kan faktisk lade sig gøre, men det er ikke helt uden problemer. For det første er der en række detaljer, som hyppigt forekommer i programmerne, der skal ændres. Det drejer sig om de PEEK- og POKE-ordrer, som refererer til helt bestemt pladser i computerens hukommelse - og disse pladser har ikke desamme numre i VIC og Commodore 64.
Løsningen på dette problem er ikke morsom. Man må sætte sig ned med en memory-map for begge computere og erstatte VIC-tallene med de tal, der gælder for samme funktion i 64'eren (eller omvendt, hvis det er et Commodore 64-program, der skal over i en VIC-20). Memory-map til C-64 findes i brugervejledningen, og for VIC kan den f.eks. skaffes gennem VI & VIC (se bestillingslisten her i bladet).
Det skal straks nævnes, at programmer, der indeholder maskinkode, også skal ændres, og her må man nok regne med massive vanskeligheder. Det er **meget svært** at skrive maskinkodeprogrammer - og det er endnu mere slidsomt at oversætte dem fra én computer til en anden.

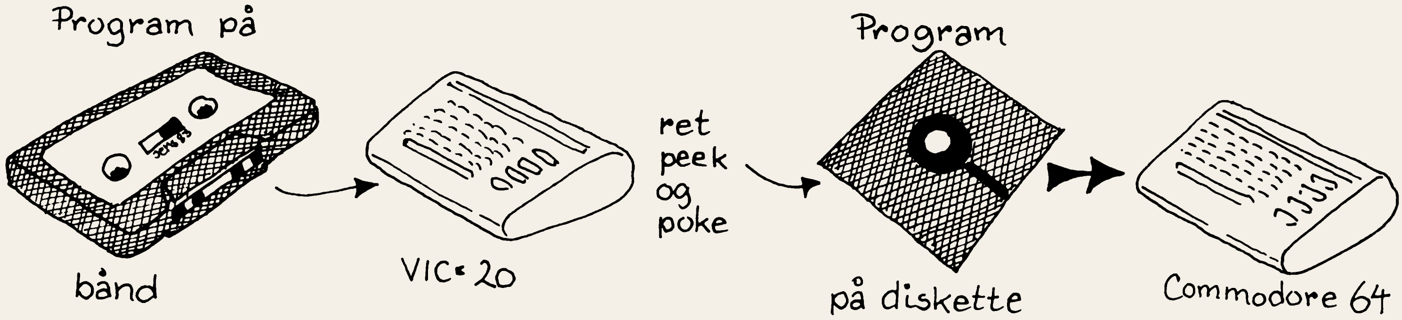
Sådan LOAD'es programmet

Men det er slet ikke sikkert, man når så langt, for der er allerede mange Commodore 64-ejere, der har måttet konstatere, at de end ikke kunne indlæse VIC-20 programmerne fra kassettebånd i deres nye computer.

Forklaringen er, at Commodore 64 har meget mere at se til, når et program indlæses - og det betyder, at TIMINGEN er meget mere kritisk, end tilfældet er for VIC-20.
Derfor er man nødt til at gå en omvej for at nå sit mål. Omvejen hedder diskettestationen - og det kan naturligvis være en ubehagelig erkendelse, hvis man ikke kan komme i nærheden af sådan en sag. Men fremgangsmåden er som følger. Programmet indlæses i VIC-20, hvorefter det lagres på diskette. Fra denne diskette skulle Commodore 64 være i stand til at læse programmet - og så kan man gå i gang med at foretage alle de ændringer, der er nødvendige for, at det kan komme til at køre.

RUN ikke

Her er det måske på sin plads at advare mod at RUN'e programmet i Commodore 64, inden det er ændret. En forkert POKE-ordre kan nemlig sende computeren i coma, så man er nødt til at slukke for den og starte helt forfra. Klogest er det, hvis man først foretager sine rettelser, derefter lagrer det redigerede program - og først derefter prøver, om det kører som det skal.



Dansk karaktersæt findes også til Commodore 64

En VIC-20 bruger, der har haft glæde af et dansk karaktersæt, er nu gået over til Commodore 64. Det giver anledning til spørgsmålet, om det danske karaktersæt til VIC-20 også kan bruges til C-64.
Det kan det ikke, men det findes tilsvarende til Commodore 64, og det repræsenterer endda en forbedring i forhold til VIC, idet Æ, Ø og Å bliver flyttet hen på de rette pladser på tastaturet - som

Om de danske karakterer har vi også fået et brev fra Verner Madsen i Rødovre. Han undrer sig over, at man ikke kan få små og store Æ, Ø og Å, men vi kan glæde ham med, at det faktisk kan lade sig gøre. Bare ikke fra tastaturet. Men tegnene findes, og de små bogstaver er CHR$-værdierne 91, 92 og 93, og de store har værdierne 219, 222 og 221 for henholdsvis Æ, Ø og Å.

Monitorprogram til 64'eren - men det kan også bruges i forbindelse med VIC-20

Måske har du fulgt med i serien omkring maskinsprog. Begreber som disassembler og assembler er afløst af ord som compiler og monitor. Dette program er et eksempel på den sidste slags: monitorprogrammet.

Her i bladet er forskellige aspekter omkring maskinprogrammering allerede belyst. Det er naturligvis sket med udgangspunkt i VIC-20's mikroprocessor. Det er den såkaldte 6502. I Commodores Commodore 64 sidder en 6510 processor. Heldigvis for os er deres maskinkodeordrer identiske. Det har meget stor betydning, når vi skal lave programmer om fra f.eks. VIC-20 til 64-modellen. Og det betyder også, at det følgende program: "Monitor*64" med ganske få ændringer - som vi skal komme tilbage til - kan løbe på begge modeller.
det er skrevet i Commodore BASIC. Det er derfor ikke særligt hurtigt, men det strækker alligevel et godt stykke. Det har alle de vigtigste ordrer, som f.eks. kendes fra VICMON. Men ikke dette programs finesser.

Programmet er godt til at lære maskinprogrammering at kende i praksis. Og det sætter dig i stand til at skrive, erstatte og gemme maskinprogrammer på 64'eren (VIC-20) i forbindelse med en recorder.

Samtidig giver det et par meget anvendelige subrutiner, som du kan bruge i andre programmer. Nu er det vist tiden, at du får mere at vide om selve programmer.

Programfunktionerne

Programmet byder på otte ordre, der giver adgang til at manipulere med indholdet af de RAM (Random Access Memory) bits, der er ladet op efter maskinens opstart.

M xxxx yyyy - viser memory fra position xxxx til yyyy

; HHHH - placerer hex-kode NN,MM,......osv. i memory fra position HHHH

G HHHH - udfører programmet fra adressen HHHH

L filenavn - lader programmet fra recorder ind i memory på den angivne absolutte adresse

S filenavn - saver maskinprogrammet på bånd

D DDDD - konverterer decimaltal op til 65535 til hex

H HHHH - konverterer hextal til decimal

X - ud af monitorprogrammet

Jeg skal nu belyse de enkelte ordrer og deres eventuelle anvendelse. De er alle ret simple, og giver derfor kun adgang til de allermest nødvendige funktioner for at få gang i maskinprogrammering. Men de virker, og vil give et godt begyndergrundlag.

M xxxx yyyy

Når man vil arbejde i maskinsprog, foregår det i det såkaldte hexidecimale talsystem, som er omtalt i tidligere numre af VI & VIC, nr. 45.10.1982). De enkelte celler i RAM indeholder - eller kan indeholde - tal fra $0 ($-tegnet angiver, at der er tale om et hex-tal) og til $ FF. Det svarer i decimaltal til 0-255. Altså 256 forskellige værdier.
Det er disse informationer, programmet giver adgang til at manipulere med. Ovenstående kode fremkalder hextallene fra memory-positionen xxxx til yyyy. Begge tal er i hex og angiver startadressen på en programlinie på ialt fire hex-tal.
Vil man f.eks. se, hvad RAM-cellerne indeholder fra position $C000 til $C019, taster man først et M så et mellemrum og $C000. Mellemrum igen og $C019 efterfulgt af et tryk på RETURN. Programmet viser nu:

```
;C000 FF 4F B0 00 FF
;C005 4F B0 00 FF 4F . . .
osv. indtil
;C019 4F B0 00 FF 4F
```

Den stump program, som er afdækket, viser sig at være en cyklus. Den befinder sig på positionen 49152 (C000), der er starten på hele 4K RAM, som er afskåret fra det område, hvor BASIC-programmet når ind.
På 4K RAM - dette gælder naturligvis kun for C 64 - kan der stå en god mundfuld maskinprogram. Nok til selv den mest kræsne nybegynder.
Vil man på VIC-20 bruge dette program, må man bruge den sædvanlige afspærringsteknik, så maskinprogrammet ikke overskrides. Eller bruge de relativt få RAM-celler, der kun bruges sjældent. $C000 kan altså bruges til at gemme maskinprogrammer i, som vi senere skal se.

; HHHH

Denne ordre udføres ved at skrive semikolon mellemrum og hex-adressen, hvor man ønsker at placere sine nye informationer. Man skriver f.eks.:

```
; C000 20 FD AE 20 9E
```

og RETURN. Det bevirker, at indholdet af denne »programlinie« nu står i RAM-positionerne $C000 og frem til $C004.
I praksis vil man kun sjældent bruge denne ordre. Det er næsten udelukkende til at foretage rettelser. I stedet vil man bruge følgende fremgangsmåde, når man vil skrive et nyt program ind over det eksisterende fra lad os sige $C000 til $C01E:

M C000 C01E-ordren bruges til at vise indholdet af RAM. Derefter bruger man C64-erens curser-funktion til at finde tilbage til den første adresses hex-indhold og **overskrive** dette med det nye indhold.

Efter hver linie trykkes RETURN. Ser man nøjere på linien, svarer indholdet jo netop til det, man ville have opbygget med »semikolon-ordren«.

På dette sted var det måske på sin plads at komme med en ADVARSEL: Det er bydende nødvendigt, at en maskinrutine afsluttes med en »return« ordre. Den er i mnemonic-kode RTS og i hex 60. Eller maskinprogrammet skal slutte med en hop-ordre til en BASIC-ordre eller andet maskinprogram. Men så skal dette program afsluttes med en RTS-ordre. Ellers kan du ikke stoppe din computer - kun slukke.
Det er derfor også altid god latin at spille sit program ind på bånd, **inden** man første gang afprøver det. Der er ALTID fejl i. Flere tusinde taster har ofte været berørt inden et program er inde - det gælder også dette program - og der vil være fejlslag imellem.

G HHHH

Når man så har puslet sig frem til en maskinkode, som man har fået anbragt i RAM, kommer det store øjeblik, hvor man skal afprøve den. Men også her er der fælder. En af de stygge er, at man har glemt, at der faktisk spørges til parametre i rutinen eller programmet. Dette program åbner mulighed for at indgive dem. Men - indrømmet - på en ret kluntet måde.
Man starter med at skrive H og mellemrum efterfulgt af den fircifrede hex-adresse f.eks. $C000 og RETURN. Maskinen svarer nu med at oplyse, hvilken adresse, man har valgt, hvilken kommando, og om der er ekstra parametre knyttet til ordren. Man bekræfter med J/N, og maskinen svarer hvis J: Er parametrene læst ind? Havde man svaret N lige før, var ordren udført fra $C000.
Hvis man på det sidste spørgsmål svarer J, udføres ordren som ovenfor. Hvis det er et N, sker der følgende: Maskinen viser to instruktioner, som man skal følge. Derefter går den ud af monitorprogrammet og er klar til, at man retter i selve programlinie 680 i basic-programmet. Ret i linie 680 og RETURN og GOTO 680 (- står der). Og det er faktisk lige, hvad man skal. Men det er VIGTIGT ikke at rette i den første del af linie 680. Der skal kun rettes fra slutningen af (BR * 4096).
I det viste eksempel, som knytter sig til et lille program i maskinsprog, som giver ordren PRINT, er der tre parametre, som skal ind adskilt med komma.
Den fulde linie skal derfor hedde:
680 SYS R+(BR*4096), 15,20,"PRØVE"
Denne linie bevirker, at der skrives PRØVE i kolonne 15 linie 20. Smart, ikke?
For at få rettelsen ind i maskinen - altså de tre parametre, må man taste RETURN, som bevirker at linie 680 i basicprogrammet nu har dette indhold. For at spare endnu en opstart forfra i programmet, kan man derefter taste GOTO 650 og programudførelsen starter, som havde man tastet G C000 efter RUN-opstarten.

L file-navn

Med denne ordre er du i stand til at lagre et maskinprogram på et ganske bestemt sted i RAM. Det forudsætter, at du har savet programmet ved hjælp af s-ordren.
Med i denne artikel - eller på det bånd, som kan købes af programmøren - er et lille maskinprogram. Hvis du bruger L filenavn på båndet, vil du få indlæst PRINT@fra $C000.
Ordren bruges på følgende måde: L mellemrum anførelsestegn ('') navnet på filen - i dette tilfælde PRINT@ - og anførelsestegn efterfulgt af RETURN.
Kan du ikke huske, hvilken file, du søger, må du bruge en »snyde-kode«. Tast L '''' og RETURN. Du får herefter indlæst det næstfølgende maskinprogram, som du kan vælge at kontrollere med M-ordren eller udføre med G-ordren.
Men er du klar over, hvilket program du søger, bruger du L »filenavn« og båndet kører indtil den har fundet programmet.

S filenavn

Denne ordre saver et program kaldet »filenavn«, f.eks. PRINTe. Du får brug for den, når du har tastet BASIC-programmet ind og har afluset det for de fejl, som der altid er efter tusindvis af indtastninger.
Ordren udføres: S mellemrum anførelsestegn ('') file navnet og anførelsestegn efterfulgt af RETURN.
Mit råd til dig er: Inden du kører løs med programmet, tag da en KOPI. Du loader simpelthen programmet, hvis du har købt det færdigt. Eller du taster det ind. SAVEer det. Retter det til og SAVEer det færdige program.
Når programmet kører uden problemer, tager du en kopi: Sluk maskinen. Load programmet. Tag kassetten ud og sæt en ny tom kassette i. SAVE »Monitor*64«, og du står nu med en arbejdskopi, hvor du kan tillade dig at begå en bommert på. Sker det, så tag en ny kopi af originalen. Men tilbage til S-kommandoen.
Når du har udført den lille stump maskinprogram og indlæst den som beskrevet under M xxxx yyyy og ; HHHH-kommandoerne, kan du SAVE den. Den lagres nu under denne specifikke file-adresse og lagres altid i de angivne RAM-celler.
Tast S ''PRINT@'' og RETURN. C-64 svarer med saver S ''PRINT@''
Noter tæller på recorder og navn på maskinprogrammet på et stykke papir og TAST (betyder rør en tilfældig tast, når instruktionen er udført som beskrevet).
Maskinen svarer nu med:
Start i hex HHHH (du svarer med f.eks. C000 og RETURN)
Slut i hex HHHH (du svarer f.eks. C019 og RETURN)
Programmet sørger nu for at indlæse netop denne del af Commodore 64's program.
Og har du indlæst - indtastet - maskinprogrammet for PRINT @, som beskrevet ovenfor, har du nu en båndkopi af programmet, som du senere kan indlæse.
På denne måde kan du indlæse en lang række ekstra BASIC ordrer, som du kan bruge med SYS-kommandoen til enhver tid efter de er indlæst. Husk eventuelle parametre. I dette tilfælde skal du skrive et linienummer og SYS 49152. Eller blot SYS 49152, hvis det ikke er i et program.

D DDDD

Ordren giver dig adgang til at få konverteret et decimaltal mellem 0 og 65535 til hex-tal. Ordren udføres med D RETURN og CMB64 svarer med: dec til hex? og du indtaster f.eks. 65535 RETURN og der svares HEX FFFF.

H HHHH

Her får du hjælp til at omsætte et hex-tal til decimal. F.eks. sådan: tast H og RETURN og Commodore 64 svarer Hex til dec? Du taster f.eks. C000 og maskinen svarer decimal 49152.

X

Endelig er der X-ordren, der kun tjener til at bringe dig ud af programmet og tilbage til programmerings-mode. Ved at taste RUN er du igen i MONITOR*64, forudsat du ikke har brugt BASIC-kommandoen NEW.

Indlæsning

Programmet er lige til at taste ind fra den viste liste. Når det er sket, eller programmet er læst ind fra bånd, SAVEs det, inden man begynder at rette. Der forekommer POKE og SYS ordre i programmet, og erde tastet forkert ind eller fodret med forkerte tal af programmet selv, kan man få svært ved at stoppe programudførslen. Derfor SAVE inden RUN første gang.
Når programmet kører og er SAVEet for sidste gang bruges RUN, og man er nu i MONITOR*64. Det første, man ser, er en præsentation og en menu. Denne menu kommer man tilbage til hvergang man taster N (Ny) i programkørslen, som vist på skærmen.

Programbeskrivelse

Der bruges i programmet flere gode BASIC-rutiner, som kan anvendes i andre programmer. Jeg skal blot nævne nogle få af dem.
Linie 950 er starten på en lille rutine, som C-64 kan bruges til at kontrollere om nogen af knapperne på recorderen er trykket ned. Er de det, vil PEEK(1) antage en værdi forskellig fra 55.
Linie 9999 giver en effektiv rutine for omsætning mellem et tal i hex og et decimaltal.

Sådan får du Monitor*64 til at køre på VIC-20

Her er nogle tips til, hvordan du får programmet MONITOR*64 til at køre på en VIC-20. Der er flere ting, der skal tages i betragtning. Og du må eksperimentere dig frem med hensyn til opstilling af en ny »menu« og andre udsagn, som er opstillet efter Commodore 64's 40 kolonner pr. linie.

Jeg skal her nøjes med at give oplysninger om de POKE- og PEEK-kommandoer, som forekommer. De er ret få og derfor overkommelige at rette.

Linie 10 erstattes med:
10 POKE 36879,8 (det giver en sort skærm).

Linie 110 erstattes af:
110 IF CM = "X" THENPRINT"shift/clr CTRL7":POKE36879,27

Linierne 315 og 320 erstattes med:
320 POKE36878,15: POKE36876,215:FORX = 1TO250:NEXT:POKE 36876,0

Linierne 674, 676, 680, 685 erstattes af:
680 SYS DC, 15, 20,"PRØVE"
685 SYS DC

Linierne 960 og 970 erstattes med:
960 IF(PEEK(37151) AND64) = 0THEN PRINT"TRYK STOP PÅ KASSETTE"
970 IF(PEEK(37151) AND64) = 0THEN970

Et praktisk eksempel

PRINT@ er en kommando, man kan støde på i programlister. Men den standard BASIC, Commodore 64 er udstyret med, har ikke denne mulighed for på en nem måde at placere et stykke tekst eller en række tegn på en ønsket position.
Vil man gøre noget sådant, har man muligheden for at bruge en række curserbevægelser indesluttet i anførelsestegn. Men det tager plads, og er ikke særlig handy.

Den efterfølgende stump maskinrutine i hexkode giver denne nye kommando som et supplement.

PRINT@ bruges således:
SYS49152,x,y, ''tekst/tegn''

Glemmer man parametrene, får man "Syntax error". X-et står for kolonne, f.eks. 15. Y-et står for linie, f.eks. 20. Og i programmet MONITOR*64 bruger vi desuden "prøve" som et eksempel på en tekst, som skal placeres.

Denne nyttige ordre kan også bruges til at slette en del af skærmens billede med. F.eks. således:

2000 FORY = 0TO15:
SYS49152,Ø,Y,"
(40 eller mindre mellemrum)
":NEXT

Her følger hexkoden, som du kan indlæse med MONITOR *64 programmets M XXXX YYYY og/eller ; HHHH ordrene:

```
C000 20 FD AE 20 9E
C005 B7 8A 48 20 FD
C00A AE 20 9E B7 68
C00F A8 18 20 F0 FF
C014 20 FD AE 4C A4
C019 AA 00 00 00 00
```

Programmet er kun anvendeligt på 64-modellen.

Her er nogle af de »mystiske« tegn i programmerne

Det kan være let nok at taste et program ind i VIC så længe det bare drejer sig om at skrive tal og bogstaver, men når de forskellige grafiske symboler skal tydes, kan der let opstå vanskeligheder.
End ikke brugervejledningen for VIC-20 indeholder en oversigt over de anvendte grafiske tegn for funktionstasterne F1-F8 sådan som de ser ud, når de forekommer på skærmen eller i en programlistning. Det er lettere at referere til disse taster, hvis man kan slippe for at spekulere på, hvilken ASCII-kode, de har.
På samme måde har det givet anledning til visse vanskeligheder, at tegnet for »CURSOR TIL HØJRE« på skærmen ses som en negativ, kantet slut-parentes. Det er den, der sidder på samme tast som semikolon. Men i mange programudskrifter får man i stedet et dansk Å - også negativt.

Forklaringen er naturligvis den enkle, at printere med danske bogstaver må udelade nogle af de eksisterende tegn, og her er det altså gået ud over den nævnte parentes - mens det engelske pund-tegn må lade pladsen til fordel for Ø, og den kantede start-parentes er erstattet med Æ.

I håb om, at det kan gøre livet lidt lettere for VIC-brugerne, gengiver vi her de tegn, der oftest giver anledning til problemer (inklusive det negative Å).

```
CURSOR OP                = "[illegible]"
CURSOR NED               = "[illegible]"
CURSOR TIL HØJRE         = "[illegible]"
CURSOR TIL VENSTRE       = "[illegible]"
HOME                     = "[illegible]"
FUNKTIONSTAST F 1        = "[illegible]"
FUNKTIONSTAST F 2        = "[illegible]"
FUNKTIONSTAST F 3        = "[illegible]"
FUNKTIONSTAST F 4        = "[illegible]"
FUNKTIONSTAST F 5        = "[illegible]"
FUNKTIONSTAST F 6        = "[illegible]"
FUNKTIONSTAST F 7        = "[illegible]"
FUNKTIONSTAST F 8        = "[illegible]"
RVS ON                   = "[illegible]"
RVS OFF                  = "[illegible]"
```

```
10 POKE53280,0:POKE53281,0
20 PRINT"■■■■■■■■■■■■■■■■■■■      ■***■  MONITOR 1.1 * 64■■■***■"
25 PRINT"■■■■■■■■COPYRIGHT: FL. LERBAEK"
30 FORX=1TO3000:NEXT
32 GOSUB950
35 GOSUB6000
40 GOSUB5000
50 CM$=MID$(IN$,2,1)
60 IFCM$="M"THENGOSUB150:GOTO400
70 IFCM$=";"THENGOSUB150:GOTO550
80 IFCM$="G"THENPRINT"■":GOTO650
90 IFCM$="L"THENPRINT"■":GOTO700
100 IFCM$="S"THENPRINT"■":GOTO1000
110 IFCM$="X"THENPRINT"■■":POKE53280,14:POKE53281,6:END
120 IFCM$="H"THENPRINT"■":GOTO9999
130 IFCM$="D"THENPRINT"■":GOTO10050
150 IFF=1THENRETURN
160 PRINT"■":RETURN
300 REM ***ERROR***
310 PRINT"■■ ERROR ■■"
315 POKE54296,15:POKE54277,190:POKE54278,248
320 POKE54273,34:POKE54272,75:POKE54276,33:FORX=1TO250:NEXT:POKE54296,0
330 GOTO40
400 REM***M-KOMMANDO***
410 F=0:ST$=MID$(IN$,4,4)
420 FI$=MID$(IN$,9,4)
425 PRINT:PRINT"MASKINKODE FRA ";ST$;" TIL ";FI$
430 HX$=ST$:GOSUB2000:S1=DC:IFERTHEN300
440 HX$=FI$:GOSUB2000:FI=DC:IFERTHEN300
450 AD=S1
455 GETA$:IFA$=" "THEN40
460 PRINTCHR$(13)">:";
470 DC=AD:GOSUB2200:PRINTHX$" ";
480 SL=AD:FL=AD+5
490 FORAL=SLTOFL-1
500 DC=PEEK(AL):GOSUB2100:PRINTHX$;:IFAL<FL-1THENPRINT" ";
510 NEXTAL
520 AD=FL:IFAD<FI+1THEN455
525 GOSUB900
528 PRINT"■M XXXX YYYY"
530 F=1:GOTO40
550 REM***;-KOMMANDO***
560 AD$=MID$(IN$,4,4)
565 PRINT:PRINT"■PLACERER HEX-KODE I ADRESSE ";AD$
566 PRINT:PRINTIN$
570 HX$=AD$:GOSUB2000:IFERTHEN300
575 AD=DC
580 FORAL=1TO5
590 HX$=MID$(IN$,6+3*AL,2):GOSUB2000
600 IFERTHEN300
610 POKEAD+AL-1,DC
620 NEXTAL
625 GOSUB900
628 PRINT"■; XXXX"
630 F=1:GOTO40
650 REM***G=KOMMANDO****
660 HX$=MID$(IN$,4,4)
665 PRINT:PRINT"UDFØRER KOMMANDO FRA ";HX$
666 PRINT:PRINTIN$
670 GOSUB2000:IFERTHEN300
674 BR=INT(DC/4096)
676 R=DC-(BR*4096)
678 GOTO850
680 SYS R+(BR*4096),15,20,"PRØVE"
681 GOSUB900
682 PRINT"■G HHHH":GOTO40
685 SYS R+(BR*4096)
686 GOSUB900
690 PRINT"■G HHHH":GOTO40
700 REM***L-KOMMANDO*****
```

```
705 PRINT:PRINT"■INDLAESER ";IN$
710 NM$="":PT=5
720 IFMID$(IN$,PT,1)<>CHR$(34)THENNM$=MID$(IN$,5,PT-4):PT=PT+1:GOTO720
740 OPEN2,1,0,NM$
745 PRINT"FUNDET ";NM$;CHR$(13)"INDLAESER "
750 INPUT#2,AD,FI
760 FORCA=ADTOFI
770 INPUT#2,BT
780 POKECA,BT
790 NEXTCA
800 CLOSE2
805 GOSUB900
806 GOSUB950
807 PRINT"■L FILE-NAVN"
810 GOTO40
850 REM***PARAMETRE***
860 PRINT"ER DER EKSTRA PARAMETRE? J/N"
870 GETPA$:IFPA$=""THEN870
872 IFPA$<>"J"ANDPA$<>"N"THEN870
874 IFPA$="N"THEN685
880 PRINT"ER PARAMETRE LAEST IND? J/N"
882 GETPA$:IFPA$=""THEN882
884 IFPA$<>"J"ANDPA$<>"N"THEN882
886 IFPA$="J"THEN680
887 PRINT"■RET I LINIE 680 OG ■RETURN■ OG GOTO 650"
888 IFPA$="N"THENLIST680
900 REM***NY/GAMMEL***
910 PRINT:PRINT"■KOMMANDO: SAMME ■TAST■ - NY ■N■"
920 GETSP$:IFSP$=""THEN920
930 IFSP$="N"THEN35
940 RETURN
950 REM***SLUK FOR RECORDER***
960 IFPEEK(1)<>55THENPRINT"TRYK ■STOP■ PAA RECORDER"
970 IFPEEK(1)<>55THEN970
980 RETURN
1000 REM***S-KOMMANDO*****
1005 PRINT:PRINT"■SAVER ";IN$
1006 PRINT"■NOTER TAELLER PAA RECORDER OG NAVN       PAA MASKINPROGRAMMET";
1007 PRINT" OG ■TAST■"
1008 GETSP$:IFSP$=""THEN1008
1010 NM$="":PT=5
1020 IFMID$(IN$,PT,1)<>CHR$(34)THENNM$=MID$(IN$,5,PT-4):PT=PT+1:GOTO1020
1030 PRINTCHR$(13)"START I HEX HHHH >";:GOSUB5010
1040 HX$=MID$(IN$,2):GOSUB2000:IFERTHEN300
1050 S1=DC
1060 PRINTCHR$(13)"SLUT I HEX HHHH  >";:GOSUB5010
1070 HX$=MID$(IN$,2):GOSUB2000:IFERTHEN300
1080 FI=DC
1090 PRINTCHR$(13):OPEN2,1,1,NM$:PRINT"SKRIVER ";NM$
1100 PRINT#2,S1
1110 PRINT#2,FI
1120 FORAD=S1TOFI
1130 PRINT#2,PEEK(AD)
1140 NEXTAD
1150 CLOSE2
1151 GOSUB950
1152 PRINT"■S FILE-NAVN"
1155 GOSUB900
1160 GOTO40
1200 REM***D-KOMMANDO******
1210 HX"="":PT=4
1215 IN$=IN$+" "
1220 IFMID$(IN$,PT,1)<>" "THENHX$=MID$(IN$,4,PT-3):PT=PT+1:GOTO1220
1230 GOSUB2000
1240 PRINTCHR$(13);DC
1250 GOTO40
1400 REM***H-KOMMANDO*******
1410 DC$="":PT=4
1415 IN$=IN$+" "
1420 IFMID$(IN$,PT,1)<>" "THENDC$=MID$(IN$,4,PT-3):PT+1:GOTO1420
1430 DC=VAL(DC$):GOSUB2200
```

```
1440 PRINTCHR$(13);HX$
1450 GOTO40
2000 REM***HEX TIL DEC*******
2010 ER=0:DC=0:FORI=0TOLEN(HX$)-1
2020 D1=ASC(MID$(HX$,LEN(HX$)-I,1))-48
2030 IFD1>22THENER=1:RETURN
2040 IFD1>9THEND1=D1-7
2050 DC=DC+D1*16↑I:NEXTI
2060 RETURN
2100 REM***DEC TIL HEX(2)*****
2110 HX$="":FORI=1TO2
2120 Q=INT(DC/16):R=DC-16*Q+48
2130 IFR>57THENR=R+7
2140 HX$=CHR$(R)+HX$:DC=Q:NEXT
2150 RETURN
2200 REM***DEC TIL HEX (4)*****
2210 HX$="":FORI=1TO4
2220 Q=INT(DC/16):R=DC-16*Q+48
2230 IFR>57THENR=R+7
2240 HX$=CHR$(R)+HX$:DC=Q:NEXT
2250 RETURN
5000 REM***INPUT FRA KEYBORD***
5005 PRINTCHR$(13)">";
5010 OPEN1,0
5020 INPUT#1,IN$
5030 CLOSE1
5040 IFLEFT$(IN$,1)<>">"THENIN$=">"+IN$
5050 RETURN
6000 REM***MENU***
6001 PRINT"FØLGENDE ORDRE ER MULIGE: (KLAR TAST)"
6010 PRINT"M XXXX YYYY - VISER MEMORY FRA X TIL Y                    I HEX"
6020 PRINT"; HHHH       - PLACERER HEX-CODE NN MM.."
6025 PRINT"               I MEMORY FRA HHHH I HEX"
6030 PRINT"G HHHH       - UDFØRER PROGRAMMET                       FRA HHHH"
6040 PRINT"L FILE-NAVN  - PROG. FRA BAAND IND                      I MEMORY"
6050 PRINT"S FILE-NAVN  - SAVE'ER PROG. TIL BAAND"
6060 PRINT"D DDDD       - DEC. TIL HEX"
6070 PRINT"H HHHH       - HEX TIL DEC."
6080 PRINT"X            - UD AF MONITOR"
6090 GETA$:IFA$=""THEN6090
7000 RETURN
9999 REM***HEX TIL DEC***
10000 PRINT:PRINT"HEX TIL DECIMAL"
10001 PRINT:PRINT"
10005 INPUT"HEX TIL DEC";H$
10007 DN=0:FORH=1TOLEN(H$)
10010 DN=DN+(ASC(MID$(H$,H,1))-48+(ASC(MID$(H$,H,1))>57)*7)*(16↑(LEN(H$)-H))
10020 NEXTH
10030 PRINT"DECIMAL ";DN
10032 GOSUB900
10035 PRINT"H - HEX TIL DECIMAL"
10040 GOTO40
10050 REM***DEC TIL HEX***
10052 PRINT:PRINT"DECIMAL TIL HEX"
10055 INPUT"DEC TIL HEX";DN
10057 IFDN>65535THEN10055
10060 N1=INT(DN/4096)
10070 D2=DN-N1*4096:N2=INT(D2/256)
10080 D2=D2-N2*256:N3=INT(D2/16)
10090 D2=D2-N3*16:N4=D2
10100 H$=CHR$(N1+48-(N1>9)*7)
10110 H$=H$+CHR$(N2+48-(N2>9)*7)
10120 H$=H$+CHR$(N3+48-(N3>9)*7)
10130 H$=H$+CHR$(N4+48-(N4>9)*7)
10140 PRINT"HEX";H$
10142 GOSUB900
10144 PRINT:PRINT"D - DECIMAL TIL HEX"
10150 GOTO40

READY.
```

Den nemme måde

Vil du ikke bruge tid på indtastning og fejlfinding med mere, kan forfatteren til programmet hjælpe. Han sælger programmet både til Commodore 64 og til VIC-20 på en C-60 kassette af god kvalitet. Prisen er 65 kroner incl. forsendelse. Send en check og din adresse til:

Flemming Lerbæk
Blåkildevej 11
5750 Ringe

Foruden MONITOR *64 indeholder kassetten PRINT @ maskinprogrammet.

VIC programmer

Lad andre få mulighed for at købe de programmer, du udvikler til VIC.

Rubrikannoncer her i bladet koster 1 kr. pr. spaltemillimeter, og for overskuelighedens skyld bør annoncerne bygges op på samme måde, som det kan ses her på siden.

Det vil sige, at annonceteksten skal indeholde oplysninger om nødvendigt udstyr, hvilket medium, progammet findes på (kassettebånd, disk eller cartridge), prisen - og naturligvis oplysninger om, hvor man kan købe programmet.

Brug kuponen til din annonce.

VIC program sælges

Programmets navn: ______________________

Beskrivelse: ______________________

____

____

____

____

____

____

____

____

____

____

____

____

Hardware: ____________ **Medium:** ____________

Pris: ______________________

Købes hos: ______________________

____

Kuponen sendes til VI & VIC, Kornager29, 7100 Vejle.

ONDSKABENS LABYRINT

Du er havnet i en stor mørk labyrint, fyldt til renden med flagermus, slanger og en masse andre ulækre ting.
Hardware: VIC-20 (3585).
Medium: Kassettebånd.
Pris: 75 kr. (check med bestillingen).
Købes hos: Kim Konrad Jensen, Ærtebjergvej 4, Tåstråp, 8410 Rønde.

VIC-kontakten

Navn: ______________

Adresse: ______________

____

Postnr.: ______________

BY: ______________

Tlf.: ______________

Bemærkninger: ______________

____

____

____

____

____

Kuponen sendes til VI & VIC, Kornager 29, 7100 Vejle.

VIC-kontakten

Stephen Laursen
Dalsø Park 118, 3500 Værløse. Tlf. 02 - 48 46 18
Bemærkninger: Også andre Commodore-maskiner har interesse.

☐

Niels Stål Pedersen
Struervej 103, 7500 Holstebro. Tlf. (07) 41 06 18.
Bemærkninger: Start af lokal VIC-klub.

☐

Michael Bødstrup Hansen
Duevej 16, 5tv., 2000 København F.
Bemærkninger: Har 32K VIC, 40/80 kort, diskettestation, FORTH. Specielt interesseret i FORTH programmering. Udveksler gerne printerudskrifter af FORTH programmer.

☐

Carsten Svendsen
Kastanie Allé 2,
8570 Trusstrup.

☐

Finn Christensen
Hjermvej 31, 7600 Struer.
Tlf. (07) 85 35 09.

☐

Lad VI & VIC hjælpe dig

VI & VIC skal være med til at du kan få mere fornøjelse af din VIC-20, og derfor indbyder vi dig til at komme med dine praktiske eller mere teoretiske spørgsmål.
Pladsen vil naturligvis sætte grænserne for, hvor mange spørgsmål, der kan blive besvaret her i bladet, men vi vil naturligvis først og fremmest lægge vægt på at besvare de spørgsmål, som optager mange VIC-brugere.
Skriv til VI & VIC på adressen Kornager 29, 7100 Vejle, hvis du har spørgsmål, du gerne vil have behandlet.

VIC-20

Hjemmecomputeren der vokser med opgaverne



❶ VIC-20 computer
Det private computersystem starter med VIC-20 - hjemmecomputeren fra Commodore. Verdens første fuldt udbyggede farvecomputer til **kr. 2.995,-**

VIC-20 kan udbygges til et enkelt eller mere avanceret system efter ønske og i takt med brugerens behov. Til privat brug. Til undervisning. Til forretningsbrug.

VIC-20 har skrivemaskinetastatur. 8 funktionstaster. 255 farvekombinationer. 64 grafiske tegn. Musikgenerator. Intern hukommelse op til 32K RAM. Tilsluttes alle TV-apparater og monitorer.

❷ VIC-1530 kassettestation
Det første skridt mange tager, når de udbygger deres VIC-computersystem. Kassettestationen kan bruges til datalager eller til overførsel af programmer. **Pris: kr. 1.035,-.**

❸ VIC-1540 diskettestation
Diskettestationen giver mulighed for det fulde udbytte af VIC-computersystemet. Lagerkapaciteten udvides til 170K bytes. Op til 4 diskettestationer kan kobles til samtidigt. Velegnet til programudvikling. Hurtig dataoverførsel. **Pris: kr. 6.495,-.**

❹ VIC-1515 printer
Rigtig papirprinter. 80 karakterer og en hastighed på 30 tegn/sek. **Pris: kr. 4.945,-.**

❺ VIC-1020 moder-modul
Dette modul er bl.a. nødvendigt for at udnytte VIC fuldt ud. Gør det muligt samtidig at tilslutte f.eks. 3 - 8 og 16K RAM, superexpander, etc. **Pris: kr. 2.280,-.**

❻ Udvidelse af hukommelse
VIC-20's hukommelseskapacitet kan udvides med 3K til 27K RAM.

❼ VICMON
Gør det muligt at programmere i maskinsprog. **Pris: kr. 400,-.**

❽ VIC-1211A Superexpander
Giver højopløst grafik og ekstra kommandoer til plotning, farvelægning, etc. **Pris: kr. 595,-.**

❾ VIC-1212 Programmers aid pack
Et godt hjælpeværktøj ved programmering. Giver 14 ekstra kommandoer. **Pris: kr. 495,-.**

❿ VIC-1311 joy stick
Til spil og lignende. **Pris: kr. 95,-.**

⓫ Brugerhåndbog
Der findes let forståelig dansk brugerhåndbog til VIC, som også på enkel og klar måde fortæller om BASIC. Hveranden måned udkommer brugerbladet VI & VIC, som kan købes i løssalg eller abonnement med VIC-nyheder.

Find nærmeste autoriserede VIC-forhandler på listen og kom ind og prøv VIC hjemmecomputeren.

Alle priser er incl. moms.

commodore COMPUTER

05 - 64 11 55 eller 01 - 88 15 05

ØERNE: Ballerup: Mic Kasseapparater, 02-655111. **Bornholm:** Krogh Hansens Radio & TV, 03-998127. **Herlev:** W. Rolf Pedersen, 02-915511. **Hillerød:** Bo-el Data, 02-253131. **Holbæk:** Papirgården, 03-433003. **København:** BN Elektronic, 01-811900 og 01-184555. CR-Commander Radio, 01-343422. L. A. Elektronik Kbh., 01-551540. Betafon Radio, 01-310273. Samfundslitteratur, 01-351942. Poly Data Microcenter, 01-420705. W. Rolf Pedersen, 01-137056. **Lyngby:** BN Elektronik, 02-881900. Polyteknisk Boghandel, 02-881488. **Maribo:** JD Totalinformation, 03-881700. **Odense:** Dansk Totalinformation, 09-126488. EDB Butikken, 09-147660. Fl. Kjerulff, 09-135480. **Otterup:** Werners Radio Elektronik, 09-823333. **Ringsted:** Ahrent Flensborg, 03-610011. **Rødovre:** Datacare, 01-705858. **Slagelse:** Vest Sjællands Mikrodata, 03-534707. **St. Heddinge:** JE Kontorcenter, 03-703332. **Svendborg:** Jens Bach Kontordata, 09-212788. **Viby Sj.:** Baagøe Radio, 02-394100. **Vordingborg:** Kontorforsyningen, 03-770170. **GRØNLAND: Godthåb:** Godthåb Elektronikservice, 21865. **Julianehåb:** Musikhuset Julianehåb, 01-438133. **JYLLAND: Aalborg:** Knud Engsig Kontorudstyr, 08-126666. H. C. Elektronic, 08-142314. **Esbjerg:** Kurt Øland, 05-125299. **Fredericia:** Semicap Data, 05-931565. **Gram:** Teknik Huset, 04-822711. **Grenå:** Djursland Data, 06-326465. **Haderslev:** H. C. Reimers, 04-523714. **Hadsten:** Hadsten Elektronik, 06-980408. **Herning:** Logic Design, 07-221300. **Hjørring:** Norad, 08-960188. **Holstebro:** Pe-Co Computer System, 07-414646. **Horsens:** Dansk Computer Center, 05-611110. **Kolding:** Hauge Rasmussen Foto, 05-520070. **Randers:** Djursland Data, 06-434200. **Ringkøbing:** LP Musik, 07-322511. **Skanderborg:** Bildata, 06-525344. **Skive:** Elektroniklageret, 07-526177. **Struer:** Helmholt Elektronik, 07-852611. **Sønderborg:** G. P. Kontorsystemer, 04-424546. **Vejle:** 2R Data, 05-831600. **Århus:** Akademisk Boghandel, 06-128844. Clemens Papirteknik, 06-133922. Editor, 06-127720.

Hedelund Marketing, Vejle